Internet Rulle Database

Linnea Damber HT-23

➔ *Presentera en kort sammanfattning och analys av projektet samt dess struktur och uppbyggnad, för- och nackdelar med olika tillvägagångssätt o.s.v. i flytande text, cirka 1 sida.*

Struktur och uppbyggnad

Innan jag påbörjade projektet startade jag upp med en enkel skiss av strukturen. Dels för att få en överblick, dels för att kunna checka av olika delar under arbetets gång. Skissen innehöll alla nivåer, från packages, databas, applikation och frontend till mappar, klasser, controllers och views samt essentiellt innehåll i desamma. Skissen var ett stort stöd i det fortsatta arbetet med uppgiften (se bild s 2).

Models

En mycket enkel modell med enbart en klass, Movie, med givna properties. Modellen utgör basen för databas-tabellen Movies (Code first), där varje egenskap blir en kolumn i tabellen. Det sker genom Object Relational Mapping via Entity Framework.

Database & Repository

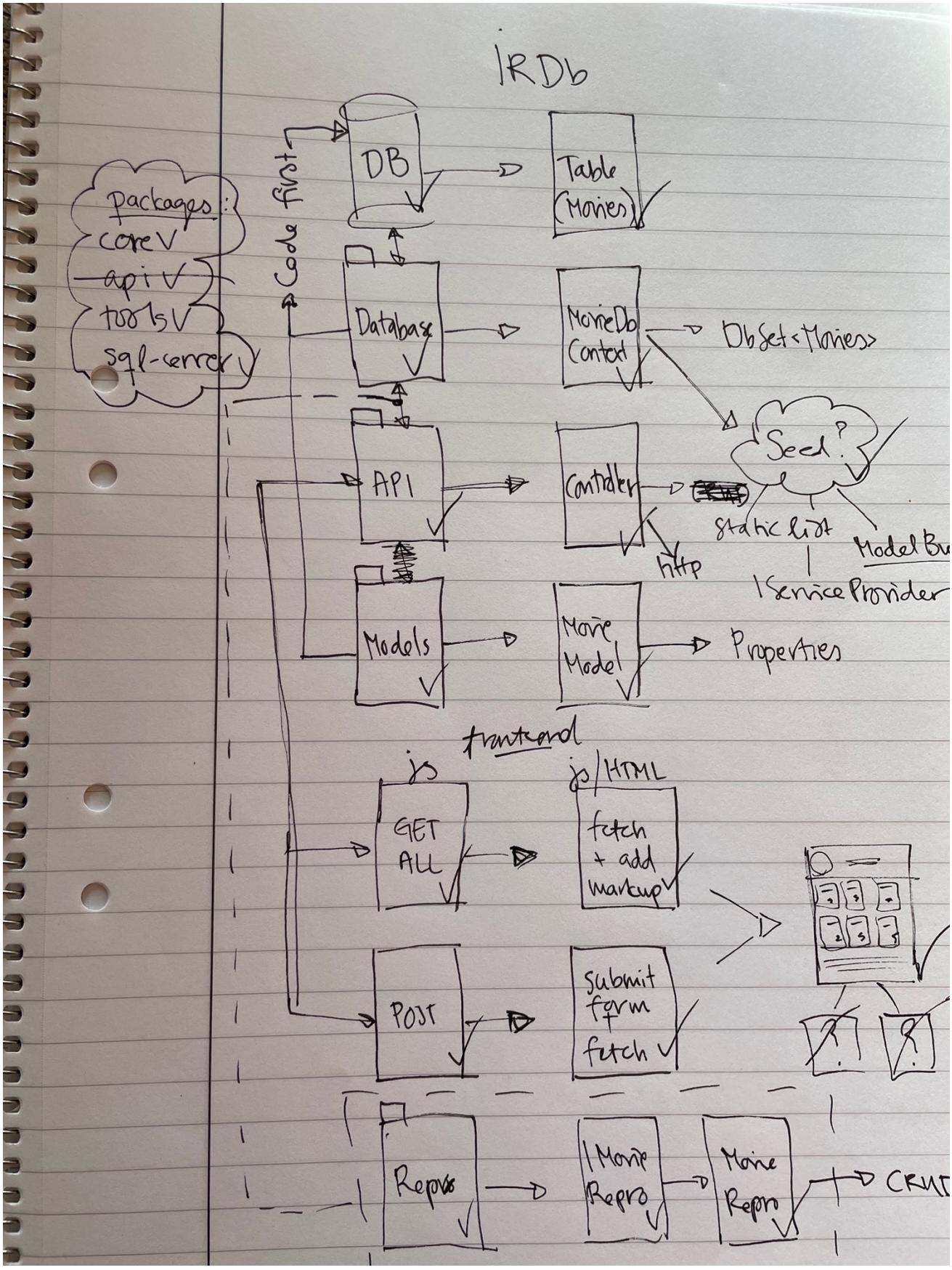
Även om vi bara hade en databastabell att arbeta med valde jag ändå att skapa ett databas-lager för min Db-context och ett Repository för att kommunicera med databasen. Det blev kanske omständligt i relation till uppgiftens omfattning, men tydligt för mig själv vilken del av applikationen som arbetar med vilken uppgift. Syftet med ett databas-lager är att samla och få en överblick över applikationens databas-kopplingar liksom Repository:n är ett sätt att samla all databaslogik på samma ställe (och enkelt kunna uppdatera den genom att implementera ett nytt Repository baserat på Interfacet). Att separera funktionalitet på detta sätt bidrar till minskade beroenden mellan olika delar, sk ”separation of concerns”. Det gör också koden lättare att underhålla över tid. Jag valde att seeda data vid initiering via ModelBuilder. Det kändes mest ”EF”.

API

Grunden i API:n är Controllern. Det är här kommunikationen sker med servern med hjälp av http-requests. Jag valde att låta mina requests return:a ActionResults. Det svåraste var att reda ut vilken del av data-kontrollen som borde ske i databasen, API:n respektive repro:t.

Frontend

Min frontend är enkel, med bara en sida där man kan se alla filmer (get all) i cards och lägga till en film i databasen via ett formulär (post). I min html har jag använt ChatGPT för att skapa och tolka Bootstrap-klasser, det hade jag inte tålamod att kolla upp varje gång, men jag har satt upp DOM-trädet, redigerat layout, css och skrivit all js-kod. Utöver get och post har jag lagt till en knapp som gör en Url baserad på filmtiteln, för att hämta filmens IMDb-sida, och en primitiv sökfunktion för filmtitlar. Hade jag haft mer tid hade jag delat upp sidans funktionalitet på undersidor och länkat emellan dem, samt implementerat full CRUD i ett admin-gränssnitt.


IRDb
Packages:
core ✓
api ✓
tools ✓
sql-server ✓
Code first
DB
Table (Movies)
Database
MovieDb Context
DbSet<Movies>
Seed?
API
Controller
static list
ModelBu
IServiceProvider
Http
Models
Movie Model
Properties
frontend
js
GET ALL
js/HTML
fetch + add markup
POST
submit form + fetch
Repos
IMovie Repo
Movie Repo
CRUD